

Differentially Private Grids for Geospatial Data

Wahbeh Qardaji, Weining Yang, Ninghui Li

*Department of Computer Science, Purdue University
305 N. University Street, West Lafayette, IN 47907, USA
{wqardaji, yang469, ninghui}@cs.purdue.edu*

Abstract— In this paper, we tackle the problem of constructing a differentially private synopsis for two-dimensional datasets such as geospatial datasets. The current state-of-the-art methods work by performing recursive binary partitioning of the data domains, and constructing a hierarchy of partitions. We show that the key challenge in partition-based synopsis methods lies in choosing the right partition granularity to balance the noise error and the non-uniformity error. We study the uniform-grid approach, which applies an equi-width grid of a certain size over the data domain and then issues independent count queries on the grid cells. This method has received no attention in the literature, probably due to the fact that no good method for choosing a grid size was known. Based on an analysis of the two kinds of errors, we propose a method for choosing the grid size. Experimental results validate our method, and show that this approach performs as well as, and often times better than, the state-of-the-art methods.

We further introduce a novel adaptive-grid method. The adaptive grid method lays a coarse-grained grid over the dataset, and then further partitions each cell according to its noisy count. Both levels of partitions are then used in answering queries over the dataset. This method exploits the need to have finer granularity partitioning over dense regions and, at the same time, coarse partitioning over sparse regions. Through extensive experiments on real-world datasets, we show that this approach consistently and significantly outperforms the uniform-grid method and other state-of-the-art methods.

I. INTRODUCTION

We interact with location-aware devices on a daily basis. Such devices range from GPS-enabled cell-phones and tablets, to navigation systems. Each device can report a multitude of location data to centralized servers. Such location information, commonly referred to as geospatial data, can have tremendous benefits if properly processed and analyzed. For many businesses, a location-based view of information can enhance business intelligence and enable smarter decision making. For many researchers, geospatial data can add an interesting dimension. Location information from cell-phones, for instance, can help in various social research that is interested in how populations settle and congregate. Furthermore, location from in-car navigation systems can help provide information on areas of common traffic congestion.

If shared, such geo-spatial data can have significant impact for research and other uses. Sharing such information, however, can have significant privacy implications. In this paper, we study the problem of releasing static geo-spatial data in a private manner. In particular, we introduce methods of releasing a synopsis of two-dimensional datasets while satisfying differential privacy.

Differential privacy [1] has recently become the defacto standard for privacy preserving data release, as it is capable of providing strong worst-case privacy guarantees. We consider two-dimensional, differentially private, synopsis methods in the following framework. Given a dataset and the two-dimensional domain that tuples in the dataset are in, we view each tuple as a point in two-dimensional space. One partitions the domain into cells, and then obtains noisy counts for each cell in a way that satisfies differential privacy. The differentially private synopsis consists of the boundaries of these cells and their noisy counts. This synopsis can then be used either for generating a synthetic dataset, or for answering queries directly.

In general, when answering queries, there are two sources of error in such differentially private synopsis methods. The first source is the noise added to satisfy differential privacy. This noise has a predefined variance and is independent of the dataset, but depends on how many cells are used to answer a query. The second source is the nature of the dataset itself. When we issue a query which only partially intersects with some cell, then we would have to estimate how many data points are in the intersected cells, assuming that the data points are distributed uniformly. The magnitude of this error depends both on the distribution of points in the dataset and on the partitioning. Our approach stems from careful examination of how these two sources of error depend on the grid size.

Several recent papers have attempted to develop such differentially private synopsis methods for two-dimensional datasets [2], [3]. These papers adapt spatial indexing methods such as quadtrees and kd-trees to provide a private description of the data distribution. These approaches can all be viewed as adapting the binary hierarchical method, which works well for 1-dimensional datasets, to the case of 2 dimensions. The emphasis is on how to perform the partitioning, and the result is a deep tree.

Somewhat surprisingly, none of the existing papers on summarizing multi-dimensional datasets compare with the simple uniform-grid method, which applies an equi-width $m \times m$ grid over the data domain and then issues independent count queries on the grid cells. We believe one reason is that the accuracy of UG is highly dependent on the grid size m , and how to choose the best grid size was not known. We propose choosing m to be $\sqrt{\frac{N\epsilon}{c}}$, where N is the number of data points, ϵ is the total privacy budget, and c is some small constant depending on the dataset. Extensive experimental results, using

4 real-world datasets of different sizes and features, validate our method of choosing m . Experimental results also suggest that setting $c = 10$ work well for datasets of different sizes and different choices of ϵ , and show that UG performs as well as, and often times better than the state-of-the-art hierarchical methods in [2], [3].

This result is somewhat surprising, as hierarchical methods have been shown to greatly outperform the equivalence of uniform-grid in 1-dimensional case [4], [5]. We thus analyze the effect of dimensionality on the effectiveness of using hierarchies.

We further introduce a novel adaptive-grid method. This method is motivated by the need to have finer granularity partitioning over dense regions and, at the same time, coarse partitioning over sparse regions. The adaptive grid method lays a coarse-grained grid over the dataset, and then further partitions each cell according to its noisy count. Both levels of partitions are then used in answering queries over the dataset. We propose methods to choose the parameters for the partitioning by careful analysis of the aforementioned sources of error. Extensive experiments validate our methods for choosing the parameters, and show that the adaptive-grid method consistently and significantly outperforms the uniform grid method and other state-of-the-art methods.

The contributions of this paper are as follows:

- 1) We identify that the key challenge in differentially private synopsis of geospatial datasets is how to choose the partition granularity to balance errors due to two sources, and propose a method for choosing grid size for the uniform grid method, based on an analysis of how the errors depend on the grid size.
- 2) We propose a novel, simple, and effective adaptive grid method, together with methods for choosing the key parameters.
- 3) We conducted extensive evaluations using 4 datasets of different sizes, including geo-spatial datasets that have not been used in differentially private data publishing literature before. Experimental results validate our methods and show that they outperform existing approaches.
- 4) We analyze why hierarchical methods do not perform well in 2-dimensional case, and predict that they would perform even worse with higher dimensions.

The rest of this paper is organized as follows. In Section II, we set the scope of the paper by formally defining the problem of publishing two dimensional datasets using differential privacy. In Section III, we discuss previous approaches and related work. We present our approach in Section IV, and present the experimental results supporting our claims in Section V. Finally, we conclude in Section VI.

II. PROBLEM DEFINITION

A. Differential Privacy

Informally, differential privacy requires that the output of a data analysis mechanism be approximately the same, even if any single tuple in the input database is arbitrarily added or removed.

Definition 1 (ϵ -Differential Privacy [1], [6]): A randomized mechanism \mathcal{A} gives ϵ -differential privacy if for any pair of neighboring datasets D and D' , and any $S \in Range(\mathcal{A})$,

$$\Pr[\mathcal{A}(D) = S] \leq e^\epsilon \cdot \Pr[\mathcal{A}(D') = S].$$

In this paper we consider two datasets D and D' to be neighbors if and only if either $D = D' + t$ or $D' = D + t$, where $D + t$ denotes the dataset resulted from adding the tuple t to the dataset D . We use $D \simeq D'$ to denote this. This protects the privacy of any single tuple, because adding or removing any single tuple results in e^ϵ -multiplicative-bounded changes in the probability distribution of the output. If any adversary can make certain inference about a tuple based on the output, then the same inference is also likely to occur even if the tuple does not appear in the dataset.

Differential privacy is composable in the sense that combining multiple mechanisms that satisfy differential privacy for $\epsilon_1, \dots, \epsilon_m$ results in a mechanism that satisfies ϵ -differential privacy for $\epsilon = \sum_i \epsilon_i$. Because of this, we refer to ϵ as the privacy budget of a privacy-preserving data analysis task. When a task involves multiple steps, each step uses a portion of ϵ so that the sum of these portions is no more than ϵ .

To compute a function g on the dataset D in a differentially privately way, one can add to $g(D)$ a random noise drawn from the Laplace distribution. The magnitude of the noise depends on GS_g , the *global sensitivity* or the L_1 sensitivity of g . Such a mechanism \mathcal{A}_g is given below:

$$\begin{aligned} \mathcal{A}_g(D) &= g(D) + \text{Lap}\left(\frac{GS_g}{\epsilon}\right) \\ \text{where } GS_g &= \max_{(D, D'): D \simeq D'} |g(D) - g(D')|, \\ \text{and } \Pr[\text{Lap}(\beta) = x] &= \frac{1}{2\beta} e^{-|x|/\beta} \end{aligned}$$

In the above, $\text{Lap}(\beta)$ denotes a random variable sampled from the Laplace distribution with scale parameter β . The variance of $\text{Lap}(\beta)$ is $2\beta^2$; hence the standard deviation of $\text{Lap}\left(\frac{GS_g}{\epsilon}\right)$ is $\sqrt{2}\frac{GS_g}{\epsilon}$.

B. Problem Definition

We consider the following problem. Given a 2-dimensional geospatial dataset D , our aim is to publish a synopsis of the dataset to accurately answer count queries over the dataset. We consider synopsis methods in the following framework. Given a dataset and the two-dimensional domain that tuples in the dataset are in, we view each tuple as a point in two-dimensional space. One partitions the domain into cells, and then obtains noisy counts for each cell in a way that satisfies differential privacy. The differentially private synopsis consists of the boundary of these cells and their noisy counts. This synopsis can then be used either for generating a synthetic dataset, or for answering queries directly.

We assume that each query specifies a rectangle in the domain, and asks for the number of data points that fall in the rectangle. Such a count query can be answered using the noisy counts for cells in the following fashion. If a cell is completely included in the query rectangle, then the noisy

count is included in the total. If a cell is partially included, then one estimates the point count in the intersection between the cell and the query rectangle, assuming that the points within the cell is distributed uniformly. For instance, if only half of the area of the cell is included in the query, then one assumes that half of the points are covered by the query.

Two Sources of Error. Under this method, there are two sources of errors when answering a query. The **noise error** is due to the fact that the counts are noisy. To satisfy differential privacy, one adds, to each cell, an independently generated noise, and these noises have the same standard deviation, which we use σ to denote. When summing up the noisy counts of q cells to answer a query, the resulting noise error is the sum of the corresponding noises. As these noises are independently generated zero-mean random variables, they cancel each other out to a certain degree. In fact, because these noises are independently generated, the variance of their sum equals the sum of their variances. Therefore, the sum has variance $q\sigma^2$, corresponding to a standard deviation of $\sqrt{q}\sigma$. That is, the noise error of a query grows linearly in \sqrt{q} . Therefore, the finer granularity one partitions the domain into, the more cells are included in a query, and the larger the noise error is.

The second source of error is caused by cells that intersect with the query rectangle, but are not contained in it. For these cells, we need to estimate how many data points are in the intersected cells assuming that the data points are distributed uniformly. This estimation will have errors when the data points are not distributed uniformly. We call this the **non-uniformity error**. The magnitude of the non-uniformity error in any intersected cell, in general, depends on the number of data points in that cell, and is bounded by it. Therefore, the finer the partition granularity, the lower the non-uniformity error.

As argued above, reducing the noise error and non-uniformity error imposes conflicting demands on the partition granularity. The main challenge of partition-based differentially private synopsis lies in how to meet this challenge and reconcile the conflicting needs of noise error and non-uniformity error.

III. PREVIOUS APPROACHES AND RELATED WORK

Differential privacy was presented in a series of papers [7], [8], [9], [6], [1] and methods of satisfying it for evaluating some function over the dataset are presented in [1], [10], [11].

Recursive Partitioning. Most approaches that directly address two-dimensional and spatial datasets use recursive partitioning [12], [3], [2]. These approaches perform a recursive binary partitioning of the data domain.

Xiao et al. [2] proposed adapting the standard spatial indexing method, KD-trees, to provide differential privacy. Nodes in a KD-tree are recursively split along some dimension. In order to minimize the non-uniformity error, Xiao et al. use the heuristic to choose the split point such that the two sub-regions are as close to uniform as possible.

Cormode et al. [3] proposed a similar approach. Instead of using a uniformity heuristic, they split the nodes along the

median of the partition dimension. The height of the tree is predetermined and the privacy budget is divided among the levels. Part of the privacy budget is used to choose the median, and part is used to obtain the noisy count. [3] also proposed combining quad-trees with noisy median-based partitioning. In the quadtree, nodes are recursively divided into four equal regions via horizontal and vertical lines through the midpoint of each range. Thus no privacy budget is needed to choose the partition point. The method that gives the best performance in [3] is a hybrid approach, which they call “KD-hybrid”. This method uses a quadtree for the first few levels of partitions, and then uses the KD-tree approach for the other levels. A number of other optimizations were also applied in KD-hybrid, including the constrained inference presented in [4], and optimized allocation of privacy budget. Their experiments indicate that “KD-hybrid” outperforms the KD-tree based approach and the approach in [2].

Qardaji and Li [12] proposed a general recursive partitioning framework for multidimensional datasets. At each level of recursion, partitioning is performed along the dimension which results in the most balanced partitioning of the data points. The balanced partitioning employed by this method has the effect of producing regions of similar size. When applied to two-dimensional datasets, this approach is very similar to building a KD-tree based on noisy median.

We experimentally compare with the state-of-the-art KD-hybrid method. In Section IV-C, we analyze the effect of dimensionality and show that hierarchical methods provide limited benefit in the 2-dimensional case.

Hierarchical Transformations. The recursive partitioning methods above essentially build a hierarchy over a representation of the data points. Several approaches have been presented in the literature to improve count queries over such hierarchies.

In [4], Hay et al. proposed the notion of constrained inference for hierarchical methods to improve accuracy for range queries. This work has been mostly developed in the context of one-dimensional datasets. Using this approach, one would arrange all queried intervals into a binary tree, where the unit-length intervals are the leaves. Count queries are then issued at all the nodes in the tree. Constrained inference exploits the consistency requirement that the parent’s count should equal the sum of all children’s counts to improve accuracy.

In [5], Xiao et al. propose the Privlet method answering histogram queries, which uses wavelet transforms. Their approach applies a Harr wavelet transform to the frequency matrix of the dataset. A Harr wavelet essentially builds a binary tree over the dataset, where each node (or “coefficient”) represents the difference between the average value of the nodes in its right subtree, and the average value of the nodes in its left subtree. The privacy budget is divided among the different levels, and the method then adds noise to each transformation coefficient proportional to its sensitivity. These coefficients are then used to regenerate an anonymized version of the dataset by applying the reverse wavelet transformation. The benefit of using wavelet transforms is that they introduce a desirable

noise canceling effect when answering range queries.

For two dimensional datasets, this method uses standard decomposition when applying the wavelet transform. Viewing the dataset as a frequency matrix, the method first applies the Harr wavelet transform on each row. The result is a vector of detail coefficients for each row. Then, using the matrix of detail coefficients as input, the method applies the transformation on the columns. Noise is then added to each cell, proportional to the sensitivity of the coefficient in that cell. To reconstruct the noisy frequency matrix, the method applies the reverse transformation on each column and then each row.

Both constrained inference and wavelet methods have been shown to be very effective at improving query accuracy in the 1-dimensional case. Our experiments show that applying them to a uniform grid provides small improvements for the 2-dimensional datasets. We note that these methods can only be applied when one has decided what are the leaf cells. When combined with the uniform grid method, it requires a method to choose the right grid size, as the performance will be poor when a wrong grid size is used. In Section V, we experimentally compare with the wavelet method.

Other related work. Blum et al. [13] proposed an approach that employs non-recursive partitioning, but their results are mostly theoretical and lack general practical applicability to the domain we are considering.

[14], [15], [16] provide methods of differentially private release which assume that the queries are known before publication. The most recent of such works by Li and Miklau [16] proposes the matrix mechanism. Given a workload of count queries, the mechanism automatically selects a different set of strategy queries to answer privately. It then uses those answers to derive answers to the original workload. Other techniques for analyzing general query workloads under differential privacy have been discussed in [17], [18]. These approaches also require the base cells to be fixed. Furthermore, they require the existence of a known set of queries, which are represented as a matrix, and then compute how to combine base cells to answer the original queries. It is unclear how to use this method when one aims at answering arbitrary range queries.

A number of approaches exist for differentially private interactive data analysis, e.g., [19], and methods of improving the accuracy of such release [20] have been suggested. In such works, however, one interacts with a privacy aware database interface rather than getting access to a synopsis of the dataset. Our approach deals with the latter.

IV. THE ADAPTIVE PARTITIONING APPROACH

In this section, we present our proposed methods.

A. The Uniform Grid Method - UG

Perhaps the simplest method one can think of is the Uniform Grid (UG) method. This approach partitions the data domain into $m \times m$ grid cells of equal size, and then obtains a noisy count for each cell. Somewhat surprisingly, none of the existing papers on summarizing multi-dimensional datasets compare with UG. We believe one reason is that the accuracy

of UG is highly dependent on the grid size m , and how to choose the best grid size was not known.

We propose the following guideline for choosing m in order to minimize the sum of the two kinds of errors presented in Section II.

Guideline 1: In order to minimize the errors due to ϵ -DP UG, the grid size should be about

$$\sqrt{\frac{N\epsilon}{c}},$$

where N is the number of data points, ϵ is the total privacy budget, and c is some small constant depending on the dataset. Our experimental results suggest that setting $c = 10$ works well for the datasets we have experimented with.

Below we present our analysis supporting this guideline. As the sensitivity of the count query is 1, the noise added for each cell follows the distribution $\text{Lap}(\frac{1}{\epsilon})$ and has a standard deviation of $\frac{\sqrt{2}}{\epsilon}$. Given an $m \times m$ grid, and a query that selects r portion of the domain (where r is the ratio of the area of the query rectangle to the area of the whole domain), about rm^2 cells are included in the query, and the total noise error thus has standard deviation of $\frac{\sqrt{2}rm^2}{\epsilon} = \frac{\sqrt{2}rm}{\epsilon}$.

The non-uniformity error is proportional to the number of data points in the cells that fall on the border of the query rectangle. For a query that selects r portion of the domain, it has four edges, whose lengths are proportional to \sqrt{r} of the domain length; thus the query's border contains on the order of \sqrt{rm} cells, which on average includes on the order of $\sqrt{rm} \times \frac{N}{m^2} = \frac{\sqrt{r}N}{m}$ data points. Assuming that the non-uniformity error on average is some portion of the total density of the cells on the query border, then the non-uniformity error is $\frac{\sqrt{r}N}{c_0 m}$ for some constant c_0 .

To minimize the two errors' sum, $\frac{\sqrt{2}rm}{\epsilon} + \frac{\sqrt{r}N}{c_0 m}$, we should set m to $\sqrt{\frac{N\epsilon}{c}}$, where $c = \sqrt{2}c_0$.

Using Guideline 1 requires knowing N , the number of data points. Obtaining a noisy estimate of N using a very small portion of the total privacy budget suffices.

The parameter c depends on the uniformity of the dataset. In the extreme case where the dataset is completely uniform, then the optimal grid size is 1×1 . That is, the best method is to obtain as accurate a total count as possible, and then any query can be fairly accurately answered by computing what fraction of the region is covered by the query. This corresponds to a large c . When a dataset is highly non-uniform, then a smaller c value is desirable. In our experiments, we observe that setting $c = 10$ gives good results across datasets of different kinds.

B. The Adaptive Grids Approach - AG

The main disadvantage of UG is that it treats all regions in the dataset equally. That is, both dense and sparse regions are partitioned in exactly the same way. This is not optimal. If a region has very few points, this method might result in over-partitioning of the region, creating a set of cells with close to zero data points. This has the effect of increasing the noise error with little reduction in the non-uniformity error. On

the other hand, if a region is very dense, this method might result in *under*-partitioning of the region. As a result, the non-uniformity error would be quite large.

Ideally, when a region is dense, we want to use finer granularity partitioning, because the non-uniformity error in this region greatly outweighs that of noise error. Similarly, when a region is sparse (having few data points), we want to use a more coarse grid there. Based on this observation, we propose an Adaptive Grids (AG) approach.

The AG approach works as follows. We first lay a coarse $m_1 \times m_1$ grid over the data domain, creating $(m_1)^2$ first-level cells, and then we issue a count query for each cell using a privacy budget $\alpha\epsilon$, where $0 < \alpha < 1$. For each cell, let N' be the noisy count of the cell, AG then partitions the cell using a grid size that is adaptively chosen based on N' , creating leaf cells. The parameter α determines how to split the privacy budget between the two levels.

Applying Constrained Inference. As discussed in Section III, constrained inference has been developed in the context of one-dimensional histograms to improve hierarchical methods [4]. The AG method produces a 2-level hierarchy. For each first-level cell, if it is further partitioned into a $m_2 \times m_2$ grid, we can perform constrained inference.

Let v be the noisy count of a first-level cell, and let $u_{1,1}, \dots, u_{m_2, m_2}$ be the noisy counts of the cells that v is further partitioned into in the second level. One can then apply constrained inference as follows. First, one obtains a more accurate count v' by taking the weighted average of v and the sum of $u_{i,j}$ such that the standard deviation of the noise error at v' is minimized.

$$v' = \frac{\alpha^2 m_2^2}{(1-\alpha)^2 + \alpha^2 m_2^2} v + \frac{(1-\alpha)^2}{(1-\alpha)^2 + \alpha^2 m_2^2} \sum u_{i,j}$$

This value is then propagated to the leaf nodes by distributing the difference among all nodes equally

$$u'_{i,j} = u_{i,j} + \left(v' - \sum u_{i,j} \right).$$

When $m_2 = 1$, the constrained inference step becomes issuing another query with budget $(1-\alpha)\epsilon$ and then computing a weighted average of the two noisy counts.

Choosing Parameters for AG. For the AG method, we need to decide the formula to adaptively determine the grid size for each first-level cell. We propose the following guideline.

Guideline 2: Given a cell with a noisy count of N' , to minimize the errors, this cell should be partitioned into $m_2 \times m_2$ cells, where m_2 is computed as follows:

$$\left\lceil \sqrt{\frac{N'(1-\alpha)\epsilon}{c_2}} \right\rceil,$$

where $(1-\alpha)\epsilon$ is the remaining privacy budget for obtaining noisy counts for leaf cells, $c_2 = c/2$, and c is the same constant as in Guideline 1.

The analysis to support this guideline is as follows. When the first-level cell is further partitioned into $m_2 \times m_2$ leaf

cells, only queries whose borders go through this first-level cell will be affected. These queries may include 0, 1, 2, \dots , up to $m_2 - 1$ rows (or columns) of leaf cells, and thus $0, m_2, 2m_2, \dots, (m_2-1)m_2$ leaf cells. When a query includes more than half of these leaf cells, constrained inference has the effect that the query is answered using the count obtained in the first level cell minus those leaf cells that are not included in the query. Therefore, on average a query is answered using

$$\frac{1}{m_2} \left(\sum_{i=0}^{m_2-1} \min(i, m_2 - i) \right) m_2 \approx \frac{(m_2)^2}{4}$$

leaf cells, and the average noise error is on the order of $\sqrt{\frac{(m_2)^2}{4} \frac{\sqrt{2}}{(1-\alpha)\epsilon}}$. The average non-uniformity error is about $\frac{N'}{c_2 m_2}$; thereby to minimize their sum, we should choose m_2 to be about $\sqrt{\frac{N'(1-\alpha)\epsilon}{\sqrt{2}c_2/2}}$.

The choice of m_1 , the grid-size for the first level, is less critical than the choice of m_2 . When m_1 is larger, the average density of each cell is smaller, and the further partitioning step will partition each cell into fewer number of cells. When m_1 is smaller, the further partitioning step will partition each cell into more cells. In general, m_1 should be less than the grid size for UG computed according to Guideline 1, since it will further partition each cell. At the same time, m_1 should not be too small either. We set

$$m_1 = \max \left(10, \frac{1}{4} \left\lceil \sqrt{\frac{N\epsilon}{c}} \right\rceil \right).$$

The choice of α also appears to be less critical. Our experiments suggest that setting α to be in the range of [0.2, 0.6] results in similar accuracy. We set $\alpha = 0.5$.

C. Comparing with Existing Approaches

We now compare our proposed UG and AG with existing hierarchical methods, in terms of runtime efficiency, simplicity, and extensibility to higher dimensional datasets.

Efficiency. The UG and AG methods are conceptually simple and easy to implement. They also work well with very large datasets that cannot fit into memory. UG can be performed by a single scan of the data points. For each data point, UG just needs to increase the counter of the cell that the data point is in by 1. AG requires two passes over the dataset. The first pass is similar to that of UG. In the second pass, it first computes which first-level cell the data point is in, and then which leaf cell it is in. It then increases the corresponding counter.

We point out that another major benefit of UG and AG over recursive partition-based methods are their higher efficiency. For all these methods, the running time is linear in the depth of the tree, as each level of the tree requires one pass over the dataset. Existing recursive partitioning methods have much deeper trees (e.g., reaching 16 levels is common for 1 million data points). Furthermore, these methods require expensive computation to choose the partition points.

Effect of Dimensionality. Existing recursive partitioning approaches can be viewed as adapting the binary hierarchical

method, which works well for 1-dimensional dataset, to the cases of 2 dimensions. Some of these methods adapt quadtree methods, which can be viewed as extending 1-dimensional binary trees to 2 dimensions. The emphasis is on how to perform the binary partition, e.g., using noisy mean, exponential method for finding the median, exponential method using non-uniformity measurement, etc. The result is a deep tree.

We observe, however, while a binary hierarchical tree works well for the 1-dimensional case, their benefit for the 2-dimensional case is quite limited, and the benefit can only decrease with higher dimensionality. When building a hierarchy, the interior of a query can be answered by higher-level nodes, but the borders of the query have to be answered using leaf nodes. The higher the dimensionality, the larger the portion of the border region.

For example, for a 1-dimensional dataset with domain divided into M cells, when one groups each b adjacent cells into one larger cell, each larger cell is of size $\frac{b}{M}$ of the whole domain. Each query has 2 border regions which need to be answered by leaf cells; each region is of size on the order of that of one larger cell, i.e., $\frac{b}{M}$ of the whole domain. In the 2-dimensional case, with a $m \times m$ grid and a total of $M = m \times m$ cells, if one groups $b = \sqrt{b} \times \sqrt{b}$ adjacent cells together, then a query's border, which needs to be answered by leaf nodes, has 4 sides, and each side is of size on the order of $\frac{\sqrt{b}}{\sqrt{M}}$ of the whole domain. Note that $4\frac{\sqrt{b}}{\sqrt{M}}$ is much larger than $2\frac{b}{M}$, since M is always much larger than b . For example, when $M = 10,000$ and $b = 4$, $4\frac{\sqrt{b}}{\sqrt{M}} = 0.08$, and $2\frac{b}{M} = 0.0008$.

Therefore, in 2-dimensional case, one benefits much less from a hierarchy, which provides less accurate counts for the leaf cells. This effect keeps growing with dimensionality. For d dimensions, the border of a query has $2d$ hyperplanes, each of size on the order of $\frac{\sqrt{b}}{\sqrt{M}}$. In our experiments, we have observed some small benefits for using hierarchies, which we conjecture will disappear with 3 or higher dimensional cases.

This analysis suggests that our approach of starting from the Uniform Grid method and trying to improve upon this method is more promising than trying to improve a hierarchical tree based method. When focusing on Uniform Grid, the emphasis in designing the algorithm shifts from choosing the axis for partitioning to choosing the partition granularity. When one partitions a cell into two sub-cells, the question of how to perform the partitioning depending on the data in the cell seems important and may affect the performance; and thus one may want to use part of the privacy budget to figure out what the best partitioning point is. On the other hand, when one needs to partition a cell into, e.g., 8×8 , sub-cells in a differentially private way, it appears that the only feasible solution is to do equi-width partition. Hence the only parameter of interest is what is the grid size.

V. EXPERIMENTAL RESULTS

A. Methodology

We have conducted extensive experiments using four real datasets, to compare the accuracy of different methods and to

validate our analysis of the choice of parameters.

Datasets. We illustrate these datasets by plotting the data points directly in Figure 1. We also present the parameters for these datasets in Table II.

The first dataset (which we call the “road” dataset) includes the GPS coordinates of road intersections in the states of Washington and New Mexico, obtained from 2006 TIGER-Line from US Census. This is the dataset that was used in [3] for experimental evaluations. There are about 1.6M data points. As illustrated in Figure 1(a), the distribution of the data points is quite unusual. There are large blank areas with two dense regions (corresponding to the two states).

The second dataset is derived from the checkin dataset¹ from the Gowalla location-based social networking website, where users share their locations by checking-in. This dataset records time and location information of check-ins made by users over the period of Feb. 2009 - Oct. 2010. We only use the location information. There are about 6.4M data points. The large size of the dataset makes it infeasible to run the implementation of KD-tree based methods obtained from authors of [3] due to memory constraints. We thus sampled 1M data points from this dataset, and we call this the “checkin” dataset. As illustrated in Figure 1(b), the shape vaguely resembles a world map, but with the more developed and/or populous areas better represented than other areas.

We obtained both the third dataset (“landmark” dataset) and the fourth dataset (“storage” dataset) from infochimps. The landmark dataset² consists of locations of landmarks in the 48 continental states in the United States. The listed landmarks range from schools and post offices to shopping centers, correctional facilities, and train stations from the 2010 Census TIGER point landmarks. There are over 870k data points. As illustrated in Figure 1(b), the dataset appears to match the population distribution in US.

The storage dataset³ includes US storage facility locations. Included are national chain storage facilities, as well as locally owned and operated facilities. This is a small dataset, consisting about 9000 data points. We chose to use this dataset to analyze whether our analysis and guideline in Section IV holds for both large and small datasets.

Absolute and Relative Error. Following [3], we primarily consider the relative error, defined as follows: For a query r , we use $A(r)$ to denote the correct answer to r . For a method \mathcal{M} and a query r , we use $Q_{\mathcal{M}}(r)$ to denote the answer to the query r when using the histogram constructed by method \mathcal{M} to answer the query r , then the relative error is defined as

$$RE_{\mathcal{M}}(r) = \frac{|Q_{\mathcal{M}}(r) - A(r)|}{\max\{A(r), \rho\}}$$

where we set ρ to be $0.001 * |D|$, where D is the total number of data points in D . This avoids dividing by 0 when $A(r) = 0$.

¹<http://snap.stanford.edu/data/loc-gowalla.html>

²<http://www.infochimps.com/datasets/storage-facilities-by-landmarks>

³<http://www.infochimps.com/datasets/storage-facilities-by-neighborhood-2>

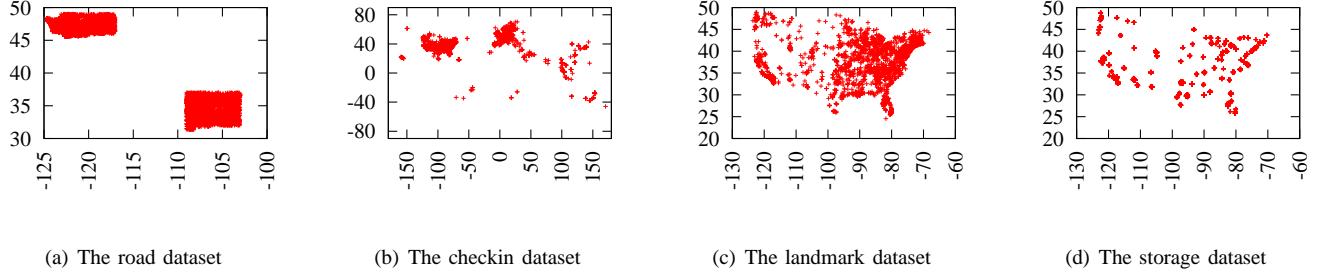


Fig. 1: Illustration of datasets.

K_{st}	KD-standard
K_{hy}	KD-hybrid
U_m	UG with $m \times m$ grid
W_m	Privlet with $m \times m$ grid
$H_{b,d}$	Hierarchy with d levels and $b \times b$ branching
A_{m_1, c_2}	AG with $m_1 \times m_1$ grid and the given c_2 value

TABLE I: Notation for Algorithms.

$RE_{\mathcal{M}}(r)$ is likely to be largest when the query r is mid-size. When the range of a query, r , is large, $RE_{\mathcal{M}}(r)$ is likely to be small since $A(r)$ is likely to be large. On the other hand, when the range of r is small, the absolute error $|Q_{\mathcal{M}}(r) - A(r)|$ is likely to be small.

While we primarily use relative error, we also use absolute error in the final comparison.

Understanding the Figures. We use two ϵ values, $\epsilon = 0.1$ and $\epsilon = 1$. For each algorithm, we use 6 query sizes, with q_1 being the smallest, each q_{i+1} doubles both the x range and y range of q_i , thereby quadrupling the query area, and q_6 , the largest query size covering between $1/4$ and $1/2$ of the whole space. The query sizes we have used are given in Table II.

For each query size, we randomly generate 200 queries, and compute the errors in answering them. We use two kinds of graphs. To illustrate the results across different query sizes, we use line graphs to plot the arithmetic mean of the relative error for each query size. To provide a clearer comparison among different algorithms, we use candlesticks to plot the profile of relative errors for all query sizes. Each candlestick provides 5 pieces of information: the 25 percentile (the bottom of candlestick), the median (the bottom of the box), the 75 percentile (the top of the box), the 95 percentile (the top of the candlestick), and the arithmetic mean (the black bar). We pay the most attention to the arithmetic mean.

Algorithm Notation. The notation for the algorithms we use in our experiments are given in Table I. The AG method is denoted by A_{m_1, c_2} , which first lays a $m_1 \times m_1$ grid, then uses $\alpha\epsilon$ to issue count query for each cell. In addition, it partitions each cell with noisy count N' into $m_2 \times m_2$ grid, with $m_2 = \left\lceil \sqrt{\frac{N'(1-\alpha)\epsilon}{c_2}} \right\rceil$. Unless explicitly noted, α is set to be 0.5.

B. Comparing KD-Tree with UG

In the first set of experiments, we compare KD-standard, KD-hybrid with UG with different grid sizes, and we identify the best performing grid size for UG. The results are presented in Figure 2.

Analysis of Results. We can observe that generally the relative errors are maximized at queries of the middle sizes. More specifically, the maximizing points are q_5 for the road dataset, q_4 for the checkin dataset, and q_3 for landmark and storage. We believe this is due to the existence of large blank areas in the road dataset and the checkin dataset. The large blank areas cause large queries to have low true count, which cause large relative errors due to the large noise error for large queries.

We can see that when varying the grid size for the UG method, there exist a range of sizes where the methods perform the best. Larger or smaller sizes tend to perform worse. When leaving the optimal range, the error steadily increases. This suggests that choosing a good grid size is important.

The ranges for the experimentally observed optimal grid sizes are give in Table II. We can see that Guideline 1 works remarkably well. The predicted best UG size generally lie within the range of the sizes that experimentally perform the best, and often fall in the middle of the range. In two cases, the predicted size lies outside the observed optimal range. For the storage dataset with $\epsilon = 1$, the predicted UG size is 30, which is quite close to 32–64, the range of the sizes observed to have lowest. Only on the road dataset (which has unusually high uniformity) at $\epsilon = 1$, our prediction (400) lies outside the observed optimal range (96–192). However, we observe that even though the high uniformity calls for a smaller optimal grid size, the performance at grid sizes 384 and 512 is quite reasonable; indeed, the average relative error in both cases are still lower than that of KD-hybrid. Jumping ahead, in Figure 6(b) we will see that U_{400} significantly outperforms U_{96} in terms of absolute error, further validating Guideline 1.

We can also see that the KD-hybrid method performs worse than the best UG method on the road dataset and the storage dataset, and is very close to the best UG method on the other two datasets.

Effect of Adding Hierarchies. In Figure 3, we evaluate the effect of adding hierarchies to UG to improve its accuracy. Our

dataset	# of points	domain size	size of q_6	size of q_1	best grid size $\epsilon = 1$			best grid size $\epsilon = 0.1$		
					UG sugg.	UG actual	AG actual	UG sugg.	UG actual	AG actual
road	1.6M	25 × 20	16 × 16	0.5 × 0.5	400	96-192	32-48	126	48-128	10-32
checkin	1M	360 × 150	192 × 96	6 × 3	316	192-384	48-96	100	64-128	16-48
landmark	0.9M	60 × 40	40 × 20	1.25 × 0.625	300	256-512	64-128	95	64-128	32-64
storage	9K	60 × 40	40 × 20	1.25 × 0.625	30	32-64	12-32	10	10-32	10-16

TABLE II: Experimental Information About Datasets.

Columns are dataset name, number of data points, domain size, the largest query size q_6 in experiments, the smallest query size q_1 , and three grid sizes each for $\epsilon = 1$ and $\epsilon = 0.1$, including the grid size suggested by Guideline 1, the range of grid sizes that perform the best in the experiments with UG, and the range of best-performing sizes for AG.

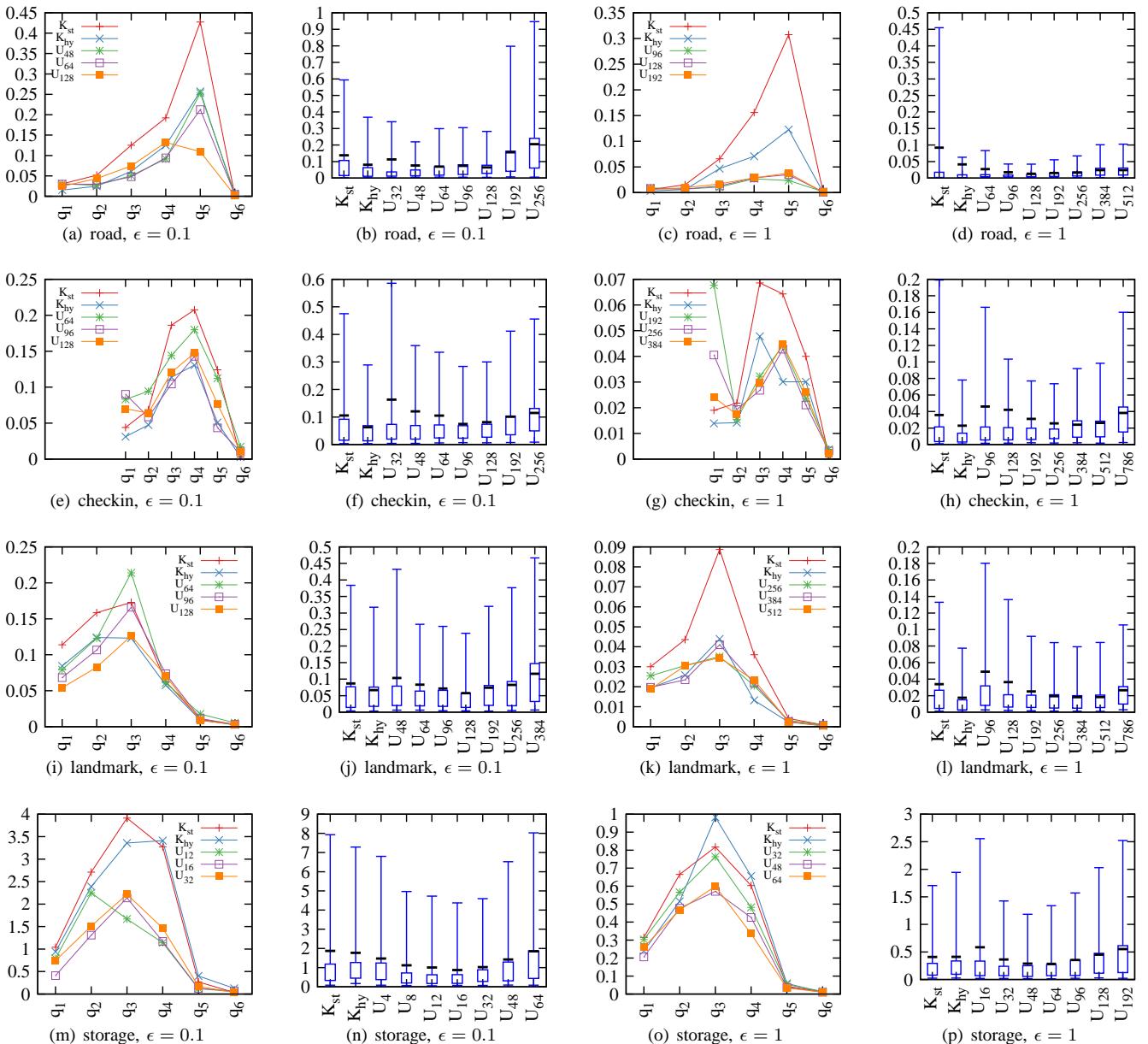


Fig. 2: Comparing KD-standard, KD-hybrid and UG with different sizes.

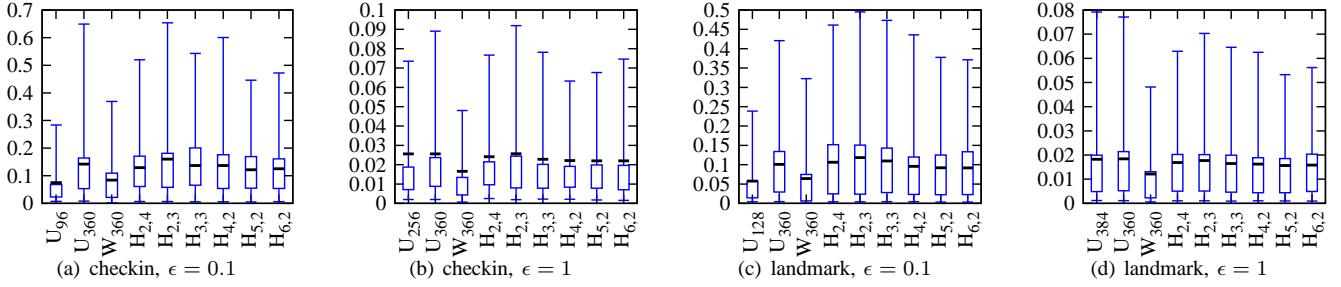


Fig. 3: Analyzing the effect of Hierarchies. In each figure, the first algorithm is UG with experimentally observed best grid size, the second is UG with 360, and the rest build hierarchies on top of a 360×360 grid. $H_{b,d}$ means build a hierarchy with a $b \times b$ branching factor and depth d .

goal is to understand whether adding hierarchies of different branching factor to UG would result in better accuracy. Here we present only results for the checkin and the landmark dataset, because the road dataset is unusual, and the storage dataset is too small to benefit from a hierarchy.

We include results for the UG method with the lowest observed relative error, the UG method with $m = 360$, which is close to the size suggested by Guideline 1 and is multiples of many numbers, facilitating experiments with different branching factors. We also include results for W_{360} , which applies the Privlet [5] method, described in Section III, to leaf cells from a 360×360 grid. We consider hierarchical methods with branching factors ranging from 2×2 to $b \times b$. We also vary depths of the tree for the branching factor 2×2 . That is $H_{2,3}$ uses 3 levels, with sizes at 360, 180, 90.

From the results we observe that while adding hierarchies can somewhat improve the accuracy, the benefit is quite small. In Section IV-C, we have analyzed the reason for this. Applying Privlet, however, results in clear, if not significant, accuracy improvements. This can be attributed to the noise reduction effect that the Privlet method has over general hierarchical methods. Jumping slightly ahead to Figure 5, we observe, however, applying Privlet to smaller grid sizes (e.g., ≤ 128) tends to be worse than the UG.

C. Evaluating Adaptive Grids

Figure 4 presents experimental results on the effect of choosing different parameters for the AG method. We use checkin and landmark datasets. The first column shows comparison of the three AG methods with best performing grid sizes with the best-performing UG method and the Privlet method with the same grid size. We show results for different query sizes. We see that the AG methods outperform UG and Privlet across all query sizes.

The second column shows the effect of varying m_1 , the first-level grid size of the AG method. We see that while the AG method like the UG method is affected by m_1 , it is less sensitive to m_1 and provides good performance for a wider range of m_1 , and that the m_1 suggested by Guideline 2 is either at or close to the optimal size.

The third and the fourth columns explore the effect of varying α and c_2 . In each figure, there are 9 candlesticks, divided into 3 groups of 3 each. The left group uses $\alpha = 0.25$, the middle group uses $\alpha = 0.5$, and the right group uses $\alpha = 0.75$. Within each group, we vary the value of c_2 . As can be seen, setting $c_2 = c/2 = 5$ as suggested significantly outperforms larger values of c_2 , namely 10 and 15. We have also conducted experiments with c_2 values from 3 to 9, and the results (which we did not include in the paper for space limitation) show that setting c_2 in the range of 3 to 7 result in almost the same accuracy. The effect of varying α can be seen by comparing the left group of 3, with the middle group, and the right group. We observe that setting $\alpha = 0.75$ performs worse than the other α values. Setting $\alpha = 0.25$ and $\alpha = 0.5$ give very similar results, perhaps with $\alpha = 0.25$ slightly better. We have also experimented with setting α from 0.1 to 0.9, with increment of 0.1. The results suggest that setting α in the range of 0.2 to 0.6 give very similar results. We use $\alpha = 0.5$ as the default value.

D. Final Comparison

In Figure 5 we perform an exhaustive comparison of 6 methods: KD-hybrid, UG with size giving lowest observed relative error, Privlet on this grid size, AG with m_1 giving lowest observed relative error, UG with suggested size, AG with suggested size. We use all 4 datasets, and two ϵ values (0.1 and 1). From these results, we observe that AG consistently and significantly outperforms other methods. We also observe that UG with the suggested grid sizes provides about the same accuracy as KD-hybrid, and AG with suggested grid sizes clearly outperforms all non-AG methods. When compared with AG with the experimentally observed best grid size, the results are slightly worse but in general quite close.

In Figure 6 we plot the same comparisons, but using absolute error, instead of relative error. Here we use logscale for the candlesticks because the ranges of the absolute errors are quite large. Again we observe that AG methods consistently and significantly outperforms other methods. It is interesting to note that for the road dataset, we observe that UG with suggested sizes outperform UG using sizes optimized for the relative error. Recall that this is the only dataset that has a large

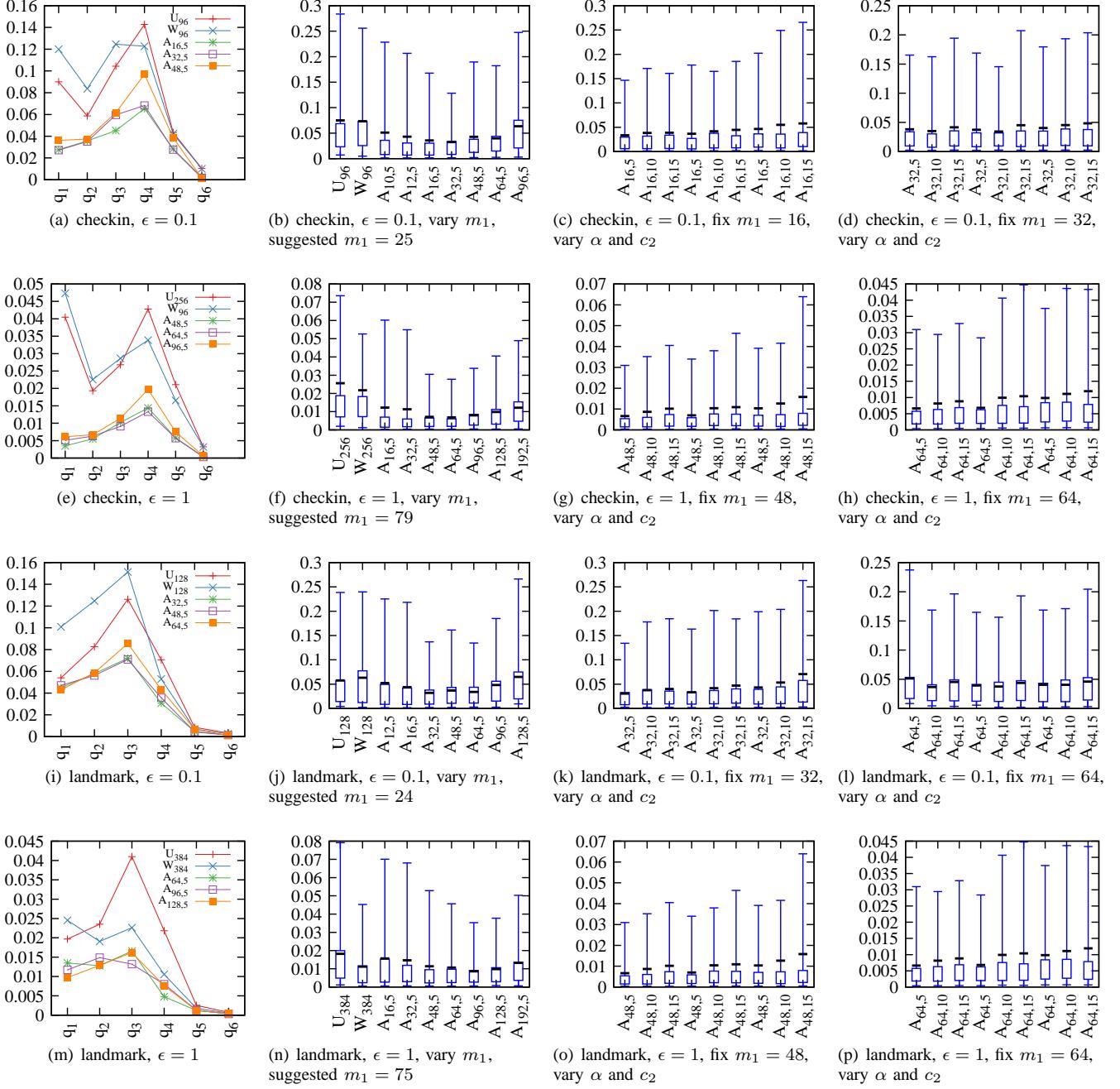


Fig. 4: Varying the parameters for the AG method. For figures in the first and second columns, $\alpha = 0.5$. For the third and fourth columns, each figure has 9 candlesticks, the left three use $\alpha = 0.25$, the middle three use $\alpha = 0.5$, and the right three use $\alpha = 0.75$.

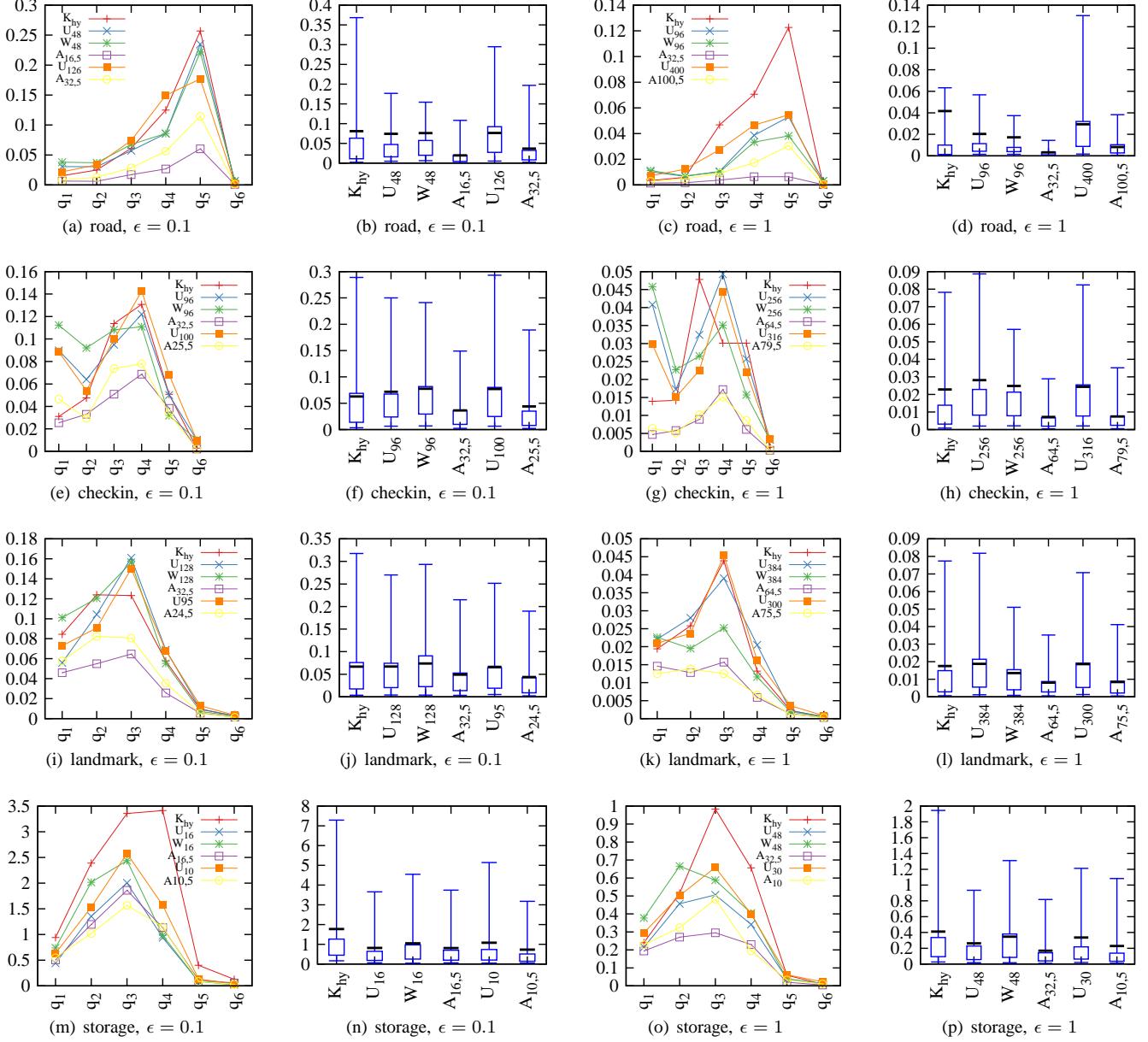


Fig. 5: Comparing, from left to right, KD-hybrid, UG with size giving lowest observed relative error, Privlet on this grid size, AG with m_1 giving lowest observed relative error, UG with suggested size, AG with suggested size.

difference between suggested size and observed optimal size, because the dataset is highly uniform. When one considers absolute errors, our suggest size seem to work very well. This suggests the robustness of our error analysis and the guidelines that follow from the analysis. Recall that our analysis did not depend upon the using of relative error or absolute error.

VI. CONCLUSION

In this paper, we tackle the problem of releasing a differentially private synopsis for two dimensional datasets. We have identified how to choose the partition granularity to balance errors due to two sources as the key challenge in differentially private synopsis methods, and propose a methodology

for choosing grid size for the uniform grid method, based on an analysis of how the errors depend on the grid size. We have proposed a novel, simple, and effective adaptive grid method, together with methods for choosing the key parameters. We have conducted extensive evaluations using 4 real datasets, including large geo-spatial datasets that have not been used in differentially private data publishing literature before. Experimental results validate our methodology and show that our methods outperform existing approaches. We have analyzed the effect of dimensionality on hierarchical methods, illustrating why hierarchical methods do not provide significant benefit in 2-dimensional case, and predicting that

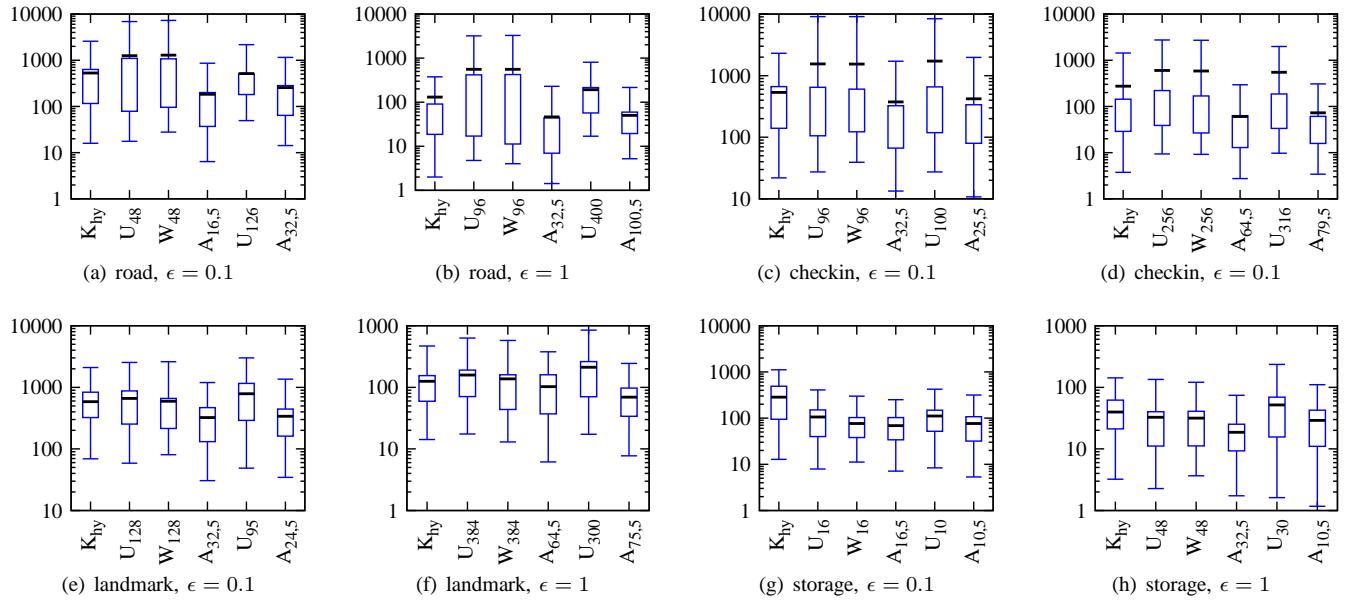


Fig. 6: Comparing the absolute error of 6 methods, from left to right, KD-hybrid, UG with size giving lowest observed relative error, Privlet on this grid size, AG with m_1 giving lowest observed relative error, UG with suggested size, AG with suggested size. Here we use log scale because the absolute error have large ranges.

they would perform even worse with higher dimensions.

REFERENCES

- [1] C. Dwork, “Differential privacy,” in *ICALP*, 2006, pp. 1–12.
- [2] Y. Xiao, L. Xiong, and C. Yuan, “Differential private data release through multidimensional partitioning,” in *VLDB SDM Workshop*, 2010.
- [3] G. Cormode, M. Procopiuc, E. Shen, D. Srivastava, and T. Yu, “Differential private spatial decompositions,” in *ICDE*, 2012.
- [4] M. Hay, V. Rastogi, G. Miklau, and D. Suciu, “Boosting the accuracy of differentially private histograms through consistency,” *Proc. VLDB Endow.*, vol. 3, pp. 1021–1032, September 2010.
- [5] X. Xiao, G. Wang, and J. Gehrke, “Differential privacy via wavelet transforms,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, pp. 1200–1214, 2011.
- [6] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *TCC*, 2006, pp. 265–284.
- [7] I. Dinur and K. Nissim, “Revealing information while preserving privacy,” in *PODS ’03: Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. New York, NY, USA: ACM, 2003, pp. 202–210.
- [8] C. Dwork and K. Nissim, “Privacy-preserving datamining on vertically partitioned databases,” in *CRYPTO*, 2004, pp. 528–544.
- [9] A. Blum, C. Dwork, F. McSherry, and K. Nissim, “Practical privacy: the sulq framework,” in *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, ser. PODS ’05. New York, NY, USA: ACM, 2005, pp. 128–138.
- [10] K. Nissim, S. Raskhodnikova, and A. Smith, “Smooth sensitivity and sampling in private data analysis,” in *STOC*, 2007, pp. 75–84.
- [11] F. McSherry and K. Talwar, “Mechanism design via differential privacy,” in *FOCS*, 2007, pp. 94–103.
- [12] W. Qardaji and N. Li, “Recursive partitioning and summarization: A general framework for privacy preserving data publishing,” in *Proceedings of the 7th ACM Symposium on Information, Computer, and Communications Security*. ACM, 2012.
- [13] A. Blum, K. Ligett, and A. Roth, “A learning theory approach to non-interactive database privacy,” in *STOC*, 2008, pp. 609–618.
- [14] C. Dwork, M. Naor, O. Reingold, G. Rothblum, and S. Vadhan, “On the complexity of differentially private data release: efficient algorithms and hardness results,” *Proceedings of the 41st annual ACM symposium on Theory of computing*, pp. 381–390, 2009.
- [15] C. Dwork, G. Rothblum, and S. Vadhan, “Boosting and differential privacy,” *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pp. 51 – 60, 2010.
- [16] C. Li and G. Miklau, “An adaptive mechanism for accurate query answering under differential privacy,” *Proc. VLDB Endow.*, vol. 5, no. 6, pp. 514–525, Feb. 2012. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2168651.2168653>
- [17] M. Hardt and K. Talwar, “On the geometry of differential privacy,” in *Proceedings of the 42nd ACM symposium on Theory of computing*, ser. STOC ’10. New York, NY, USA: ACM, 2010, pp. 705–714.
- [18] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor, “Optimizing linear counting queries under differential privacy,” in *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, ser. PODS ’10. New York, NY, USA: ACM, 2010, pp. 123–134.
- [19] F. McSherry, “Privacy integrated queries: an extensible platform for privacy-preserving data analysis,” in *SIGMOD*, 2009, pp. 19–30.
- [20] A. Roth and T. Roughgarden, “Interactive privacy via the median mechanism,” in *Proceedings of the 42nd ACM symposium on Theory of computing*, ser. STOC ’10. New York, NY, USA: ACM, 2010, pp. 765–774.